One Millisecond Face Alignment with an Ensemble of Regression Trees

Conference Paper · June 2014

DOI: 10.13140/2.1.1212.2243

2 authors, including:

Vahid Kazemi
KTH Royal Institute of Technology
5 PUBLICATIONS **1,388** CITATIONS

One Millisecond Face Alignment with an Ensemble of Regression Trees

Vahid Kazemi and Josephine Sullivan
KTH, Royal Institute of Technology
Computer Vision and Active Perception Lab
Teknikringen 14, Stockholm, Sweden
{vahidk,sullivan}@csc.kth.se

Abstract

This paper addresses the problem of Face Alignment for a single image. We show how an ensemble of regression trees can be used to estimate the face's landmark positions directly from a sparse subset of pixel intensities, achieving super-realtime performance with high quality predictions. We present a general framework based on gradient boosting for learning an ensemble of regression trees that optimizes the sum of square error loss and naturally handles missing or partially labelled data. We show how using appropriate priors exploiting the structure of image data helps with efficient feature selection. Different regularization strategies and its importance to combat overfitting are also investigated. In addition, we analyse the effect of the quantity of training data on the accuracy of the predictions and explore the effect of data augmentation using synthesized data.

Figure 1. Selected results on the HELEN dataset. An ensemble of randomized regression trees is used to detect 194 landmarks on face from a single image in a millisecond.

1. Introduction

In this paper we present a new algorithm that performs face alignment in milliseconds and achieves accuracy superior or comparable to state-of-the-art methods on standard datasets. The speed gains over previous methods is a consequence of identifying the essential components of prior face alignment algorithms and then incorporating them in a streamlined formulation into a cascade of high capacity regression functions learnt via gradient boosting.

We show, as others have [8, 2], that face alignment can be solved with a cascade of regression functions. In our case each regression function in the cascade efficiently estimates the shape from an initial estimate and the intensities of a sparse set of pixels indexed relative to this initial estimate. Our work builds on the large amount of research over the last decade that has resulted in significant progress for face alignment [9, 4, 13, 7, 15, 1, 16, 18, 3, 6, 19]. In particular, we incorporate into our learnt regression functions two key elements that are present in several of the successful algorithms cited and we detail these elements now.

The first revolves around the indexing of pixel intensities relative to the current estimate of the shape. The extracted features in the vector representation of a face image can greatly vary due to both shape deformation and nuisance factors such as changes in illumination conditions. This makes accurate shape estimation using these features difficult. The dilemma is that we need reliable features to accurately predict the shape, and on the other hand we need an accurate estimate of the shape to extract reliable features. Previous work [4, 9, 5, 8] as well as this work, use an iterative approach (the cascade) to deal with this problem. Instead of regressing the shape parameters based on features extracted in the global coordinate system of the image, the image is transformed to a normalized coordinate system based on a current estimate of the shape, and then the features are extracted to predict an update vector for the shape parameters. This process is usually repeated several times until convergence.

The second considers how to combat the difficulty of the

inference/prediction problem. At test time, an alignment algorithm has to estimate the shape, a high dimensional vector, that best agrees with the image data and our model of shape. The problem is non-convex with many local optima. Successful algorithms [4, 9] handle this problem by assuming the estimated shape must lie in a linear subspace, which can be discovered, for example, by finding the principal components of the training shapes. This assumption greatly reduces the number of potential shapes considered during inference and can help to avoid local optima. Recent work [8, 11, 2] uses the fact that a certain class of regressors are guaranteed to produce predictions that lie in a linear subspace defined by the training shapes and there is no need for additional constraints. Crucially, our regression functions have these two elements.

Allied to these two factors is our efficient regression function learning. We optimize an appropriate loss function and perform feature selection in a data-driven manner. In particular, we learn each regressor via gradient boosting [10] with a squared error loss function, the same loss function we want to minimize at test time. The sparse pixel set, used as the regressor's input, is selected via a combination of the gradient boosting algorithm and a prior probability on the distance between pairs of input pixels. The prior distribution allows the boosting algorithm to efficiently explore a large number of relevant features. The result is a cascade of regressors that can localize the facial landmarks when initialized with the mean face pose.

The major contributions of this paper are

1. A novel method for alignment based on ensemble of regression trees that performs shape invariant feature selection while minimizing the same loss function during training time as we want to minimize at test time.

2. We present a natural extension of our method that handles missing or uncertain labels.

3. Quantitative and qualitative results are presented that confirm that our method produces high quality predictions while being much more efficient than the best previous method (Figure 1).

4. The effect of quantity of training data, use of partially labeled data and synthesized data on quality of predictions are analyzed.

2. Method

This paper presents an algorithm to precisely estimate the position of facial landmarks in a computationally efficient way. Similar to previous works [8, 2] our proposed method utilizes a cascade of regressors. In the rest of this section we describe the details of the form of the individual components of the cascade and how we perform training.

2.1. The cascade of regressors

To begin we introduce some notation. Let $\mathbf{x}_i \in \mathbb{R}^2$ be the x, y-coordinates of the ith facial landmark in an image I. Then the vector $\mathbf{S} = (\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_p^T)^T \in \mathbb{R}^{2p}$ denotes the coordinates of all the p facial landmarks in I. Frequently, in this paper we refer to the vector $\mathbf{S}$ as the shape. We use $\hat{\mathbf{S}}^{(t)}$ to denote our current estimate of $\mathbf{S}$. Each regressor, $r_t(\cdot, \cdot)$, in the cascade predicts an update vector from the image and $\hat{\mathbf{S}}^{(t)}$ that is added to the current shape estimate $\hat{\mathbf{S}}^{(t)}$ to improve the estimate:

$$\hat{\mathbf{S}}^{(t+1)} = \hat{\mathbf{S}}^{(t)} + r_t(I, \hat{\mathbf{S}}^{(t)}) \tag{1}$$

The critical point of the cascade is that the regressor r_t makes its predictions based on features, such as pixel intensity values, computed from I and indexed relative to the current shape estimate $\hat{\mathbf{S}}^{(t)}$. This introduces some form of geometric invariance into the process and as the cascade proceeds one can be more certain that a precise semantic location on the face is being indexed. Later we describe how this indexing is performed.

Note that the range of outputs expanded by the ensemble is ensured to lie in a linear subspace of training data if the initial estimate $\hat{\mathbf{S}}^{(0)}$ belongs to this space. We therefore do not need to enforce additional constraints on the predictions which greatly simplifies our method. The initial shape can simply be chosen as the mean shape of the training data centered and scaled according to the bounding box output of a generic face detector.

To train each r_t we use the gradient tree boosting algorithm with a sum of square error loss as described in [10]. We now give the explicit details of this process.

2.2. Learning each regressor in the cascade

Assume we have training data $(I_1, \mathbf{S}_1), \ldots, (I_n, \mathbf{S}_n)$ where each I_i is a face image and $\mathbf{S}_i$ its shape vector. To learn the first regression function r_0 in the cascade we create from our training data triplets of a face image, an initial shape estimate and the target update step, that is, $(I_{\pi_i}, \hat{\mathbf{S}}_i^{(0)}, \Delta\mathbf{S}_i^{(0)})$ where

$$\pi_i \in \{1, \ldots, n\} \tag{2}$$

$$\hat{\mathbf{S}}_i^{(0)} \in \{\mathbf{S}_1, \ldots, \mathbf{S}_n\} \backslash \mathbf{S}_{\pi_i} \quad \text{and} \tag{3}$$

$$\Delta\mathbf{S}_i^{(0)} = \mathbf{S}_{\pi_i} - \hat{\mathbf{S}}_i^{(0)} \tag{4}$$

for $i = 1, \ldots, N$. We set the total number of these triplets to $N = nR$ where R is the number of initializations used per image I_i. Each initial shape estimate for an image is sampled uniformly from $\{\mathbf{S}_1, \ldots, \mathbf{S}_n\}$ without replacement.

From this data we learn the regression function r_0 (see algorithm 1), using gradient tree boosting with a sum of square error loss. The set of training triplets is then updated

to provide the training data, $(I_{\pi_i}, \hat{\mathbf{S}}_i^{(1)}, \Delta\mathbf{S}_i^{(1)})$, for the next regressor r_1 in the cascade by setting (with $t = 0$)

$$\hat{\mathbf{S}}_i^{(t+1)} = \hat{\mathbf{S}}_i^{(t)} + r_t(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}) \tag{5}$$

$$\Delta\mathbf{S}_i^{(t+1)} = \mathbf{S}_{\pi_i} - \hat{\mathbf{S}}_i^{(t+1)} \tag{6}$$

This process is iterated until a cascade of T regressors $r_0, r_1, \ldots, r_{T-1}$ are learnt which when combined give a sufficient level of accuracy.

As stated each regressor r_t is learned using the gradient boosting tree algorithm. It should be remembered that a square error loss is used and the residuals computed in the innermost loop correspond to the gradient of this loss function evaluated at each training sample. Included in the statement of the algorithm is a learning rate parameter $0 < \nu \leq 1$ also known as the shrinkage factor. Setting $\nu < 1$ helps combat over-fitting and usually results in regressors which generalize much better than those learnt with $\nu = 1$ [10].

Algorithm 1 Learning r_t in the cascade

Have training data $\{(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}, \Delta\mathbf{S}_i^{(t)})\}_{i=1}^N$ and the learning rate (shrinkage factor) $0 < \nu < 1$

1. Initialise

$$f_0(I, \hat{\mathbf{S}}^{(t)}) = \arg\min_{\boldsymbol{\gamma} \in \mathbb{R}^{2p}} \sum_{i=1}^{N} \|\Delta\mathbf{S}_i^{(t)} - \boldsymbol{\gamma}\|^2$$

2. for $k = 1, \ldots, K$:
   (a) Set for $i = 1, \ldots, N$

$$\mathbf{r}_{ik} = \Delta\mathbf{S}_i^{(t)} - f_{k-1}(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)})$$

   (b) Fit a regression tree to the targets $\mathbf{r}_{ik}$ giving a weak regression function $g_k(I, \hat{\mathbf{S}}^{(t)})$.
   (c) Update

$$f_k(I, \hat{\mathbf{S}}^{(t)}) = f_{k-1}(I, \hat{\mathbf{S}}^{(t)}) + \nu\, g_k(I, \hat{\mathbf{S}}^{(t)})$$

3. Output $r_t(I, \hat{\mathbf{S}}^{(t)}) = f_K(I, \hat{\mathbf{S}}^{(t)})$

2.3. Tree based regressor

The core of each regression function r_t is the tree based regressors fit to the residual targets during the gradient boosting algorithm. We now review the most important implementation details for training each regression tree.

2.3.1 Shape invariant split tests

At each split node in the regression tree we make a decision based on thresholding the difference between the intensities of two pixels. The pixels used in the test are at positions $\mathbf{u}$ and $\mathbf{v}$ when defined in the coordinate system of the mean shape. For a face image with an arbitrary shape, we would like to index the points that have the same position relative to its shape as $\mathbf{u}$ and $\mathbf{v}$ have to the mean shape. To achieve this, the image can be warped to the mean shape based on the current shape estimate before extracting the features. Since we only use a very sparse representation of the image, it is much more efficient to warp the location of points as opposed to the whole image. Furthermore, a crude approximation of warping can be done using only a global similarity transform in addition to local translations as suggested by [2].

The precise details are as follows. Let $k_{\mathbf{u}}$ be the index of the facial landmark in the mean shape that is closest to $\mathbf{u}$ and define its offset from $\mathbf{u}$ as

$$\delta\mathbf{x}_{\mathbf{u}} = \mathbf{u} - \bar{\mathbf{x}}_{k_{\mathbf{u}}}$$

Then for a shape $\mathbf{S}_i$ defined in image I_i, the position in I_i that is qualitatively similar to $\mathbf{u}$ in the mean shape image is given by

$$\mathbf{u}' = \mathbf{x}_{i,k_{\mathbf{u}}} + \frac{1}{s_i} R_i^T \delta\mathbf{x}_{\mathbf{u}} \tag{7}$$

where s_i and R_i are the scale and rotation matrix of the similarity transform which transforms $\mathbf{S}_i$ to $\bar{\mathbf{S}}$, the mean shape. The scale and rotation are found to minimize

$$\sum_{j=1}^{p} \|\bar{\mathbf{x}}_j - (s_i R_i\, \mathbf{x}_{i,j} + \mathbf{t}_i)\|^2 \tag{8}$$

the sum of squares between the mean shape's facial landmark points, $\bar{\mathbf{x}}_j$'s, and those of the warped shape. $\mathbf{v}'$ is similarly defined. Formally each split is a decision involving 3 parameters $\boldsymbol{\theta} = (\tau, \mathbf{u}, \mathbf{v})$ and is applied to each training and test example as

$$h(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}, \boldsymbol{\theta}) = \begin{cases} 1 & I_{\pi_i}(\mathbf{u}') - I_{\pi_i}(\mathbf{v}') > \tau \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $\mathbf{u}'$ and $\mathbf{v}'$ are defined using the scale and rotation matrix which best warp $\hat{\mathbf{S}}_i^{(t)}$ to $\bar{\mathbf{S}}$ according to equation (7).

In practice the assignments and local translations are determined during the training phase. Calculating the similarity transform, at test time the most computationally expensive part of this process, is only done once at each level of the cascade.

2.3.2 Choosing the node splits

For each regression tree, we approximate the underlying function with a piecewise constant function where a constant vector is fit to each leaf node. To train the regression tree we randomly generate a set of candidate splits, that is $\boldsymbol{\theta}$'s, at each node. We then greedily choose the $\boldsymbol{\theta}^*$, from these candidates, which minimizes the sum of square error. If $\mathcal{Q}$ is the set of the indices of the training examples at a node, this corresponds to minimizing

$$E(\mathcal{Q}, \boldsymbol{\theta}) = \sum_{s \in \{l,r\}} \sum_{i \in \mathcal{Q}_{\boldsymbol{\theta},s}} \|\mathbf{r}_i - \boldsymbol{\mu}_{\boldsymbol{\theta},s}\|^2 \qquad (10)$$

where $\mathcal{Q}_{\boldsymbol{\theta},l}$ is the indices of the examples that are sent to the left node due to the decision induced by $\boldsymbol{\theta}$, $\mathbf{r}_i$ is the vector of all the residuals computed for image i in the gradient boosting algorithm and

$$\boldsymbol{\mu}_{\boldsymbol{\theta},s} = \frac{1}{|\mathcal{Q}_{\boldsymbol{\theta},s}|} \sum_{i \in \mathcal{Q}_{\boldsymbol{\theta},s}} \mathbf{r}_i, \quad \text{for } s \in \{l,r\} \qquad (11)$$

The optimal split can be found very efficiently because if one rearranges equation (10) and omits the factors not dependent on $\boldsymbol{\theta}$ then one can see that

$$\arg\min_{\boldsymbol{\theta}} E(\mathcal{Q}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{s \in \{l,r\}} |\mathcal{Q}_{\boldsymbol{\theta},s}| \, \boldsymbol{\mu}_{\boldsymbol{\theta},s}^T \, \boldsymbol{\mu}_{\boldsymbol{\theta},s}$$

Here we only need to compute $\boldsymbol{\mu}_{\boldsymbol{\theta},l}$ when evaluating different $\boldsymbol{\theta}$'s, as $\boldsymbol{\mu}_{\boldsymbol{\theta},r}$ can be calculated from the average of the targets at the parent node $\boldsymbol{\mu}$ and $\boldsymbol{\mu}_{\boldsymbol{\theta},l}$ as follows

$$\boldsymbol{\mu}_{\boldsymbol{\theta},r} = \frac{|\mathcal{Q}|\boldsymbol{\mu} - |\mathcal{Q}_{\boldsymbol{\theta},l}|\boldsymbol{\mu}_{\boldsymbol{\theta},l}}{\mathcal{Q}_{\boldsymbol{\theta},r}}$$

2.3.3 Feature selection

The decision at each node is based on thresholding the difference of intensity values at a pair of pixels. This is a rather simple test, but it is much more powerful than single intensity thresholding because of its relative insensitivity to changes in global lighting. Unfortunately, the drawback of using pixel differences is the number of potential split (feature) candidates is quadratic in the number of pixels in the mean image. This makes it difficult to find good $\boldsymbol{\theta}$'s without searching over a very large number of them. However, this limiting factor can be eased, to some extent, by taking the structure of image data into account. We introduce an exponential prior

$$P(\mathbf{u}, \mathbf{v}) \propto e^{-\lambda \|\mathbf{u}-\mathbf{v}\|} \qquad (12)$$

over the distance between the pixels used in a split to encourage closer pixel pairs to be chosen.

We found using this simple prior reduces the prediction error on a number of face datasets. Figure 4 compares the features selected with and without this prior, where the size of the feature pool is fixed to 20 in both cases.

2.4. Handling missing labels

The objective of equation (10) can be easily extended to handle the case where some of the landmarks are not labeled in some of the training images (or we have a measure of uncertainty for each landmark). Introduce variables $w_{i,j} \in [0,1]$ for each training image i and each landmark j. Setting $w_{i,j}$ to 0 indicates that the landmark j is not labeled in the ith image while setting it to 1 indicates that it is. Then equation (10) can be updated to

$$E(\mathcal{Q}, \boldsymbol{\theta}) = \sum_{s \in \{l,r\}} \sum_{i \in \mathcal{Q}_{\boldsymbol{\theta},s}} (\mathbf{r}_i - \boldsymbol{\mu}_{\boldsymbol{\theta},s})^T W_i (\mathbf{r}_i - \boldsymbol{\mu}_{\boldsymbol{\theta},s})$$

where W_i is a diagonal matrix with the vector $(w_{i1}, w_{i1}, w_{i2}, w_{i2}, \ldots, w_{ip}, w_{ip})^T$ on its diagonal and

$$\boldsymbol{\mu}_{\boldsymbol{\theta},s} = \left(\sum_{i \in \mathcal{Q}_{\boldsymbol{\theta},s}} W_i \right)^{-1} \sum_{i \in \mathcal{Q}_{\boldsymbol{\theta},s}} W_i \, \mathbf{r}_i, \quad \text{for } s \in \{l,r\} \qquad (13)$$

The gradient boosting algorithm must also be modified to account of these weight factors. This can be done simply by initializing the ensemble model with the weighted average of targets, and fitting regression trees to the weighted residuals in algorithm 1 as follows

$$\mathbf{r}_{ik} = W_i(\Delta \mathbf{S}_i^{(t)} - f_{k-1}(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)})) \qquad (14)$$

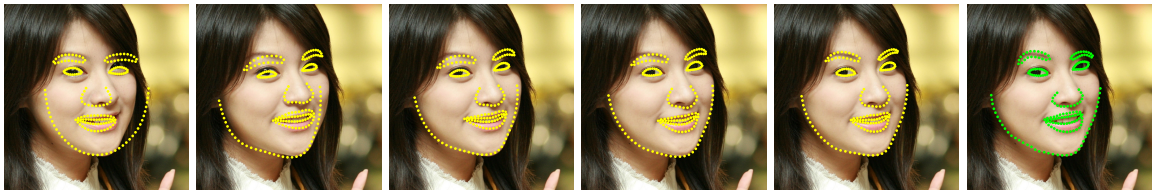
3. Experiments

Baselines: To accurately benchmark the performance of our proposed method, an ensemble of regression trees (*ERT*) we created two more baselines. The first is based on randomized ferns with random feature selection (*EF*) and the other is a more advanced version of this with correlation based feature selection (*EF+CB*) which is our reimplementation of [2]. All the parameters are fixed for all three approaches.

EF uses a straightforward implementation of randomized ferns as the weak regressors within the ensemble and is the fastest to train. We use the same shrinkage method as suggested by [2] to regularize the ferns.

EF+CB uses a correlation based feature selection method that projects the target outputs, $\mathbf{r}_i$'s, onto a random direction, $\mathbf{w}$, and chooses the pairs of features $(\mathbf{u}, \mathbf{v})$ s.t. $I_i(\mathbf{u}') - I_i(\mathbf{v}')$ has the highest sample correlation over the training data with the projected targets $\mathbf{w}^T\mathbf{r}_i$.

Parameters: Unless specified, all the experiments are performed with the following fixed parameter settings. The number of strong regressors, r_t, in the cascade is $T = 10$ and each r_t comprises of $K = 500$ weak regressors g_k. The depth of the trees (or ferns) used to represent g_k is set to $F = 5$. At each level of the cascade $P = 400$ pixel locations are sampled from the image. To train the weak regressors, we randomly sample a pair of these P pixel locations



(a) $T = 0$ (b) $T = 1$ (c) $T = 2$ (d) $T = 3$ (e) $T = 10$ (f) Ground truth

Figure 2. Landmark estimates at different levels of the cascade initialized with the mean shape centered at the output of a basic Viola & Jones[17] face detector. After the first level of the cascade, the error is already greatly reduced.

according to our prior and choose a random threshold to create a potential split as described in equation (9). The best split is then found by repeating this process $S = 20$ times, and choosing the one that optimizes our objective. To create the training data to learn our model we use $R = 20$ different initializations for each training example.

Performance: The runtime complexity of the algorithm on a single image is constant $O(TKF)$. The complexity of the training time depends linearly on the number of training data $O(NDTKFS)$ where N is the number of training data and D is dimension of the targets. In practice with a single CPU our algorithm takes about an hour to train on the HELEN[12] dataset and at runtime it only takes about one millisecond per image.

Database: Most of the experimental results reported are for the HELEN[12] face database which we found to be the most challenging publicly available dataset. It consists of a total of 2330 images, each of which is annotated with 194 landmarks. As suggested by the authors we use 2000 images for training data and the rest for testing.

We also report final results on the popular LFPW[1] database which consists of 1432 images. Unfortunately, we could only download 778 training images and 216 valid test images which makes our results not directly comparable to those previously reported on this dataset.

Comparison: Table 1 is a summary of our results compared to previous algorithms. In addition to our baselines, we have also compared our results with two variations of Active Shape Models, *STASM*[14] and *CompASM*[12].

	[14]	[12]	EF	EF+CB	EF+CB (5)	EF+CB (10)	ERT
Error	.111	.091	.069	.062	.059	.055	**.049**

Table 1. A summary of the results of different algorithms on the HELEN dataset. The error is the average normalized distance of each landmark to its ground truth position. The distances are normalized by dividing by the interocular distance. The number within the bracket represents the number of times the regression algorithm was run with a random initialization. If no number is displayed then the method was initialized with the mean shape. In the case of multiple estimations the median of the estimates was chosen as the final estimate for the landmark.

The ensemble of regression trees described in this paper significantly improves the results over the ensemble of ferns. Figure 3 shows the average error at different levels of the cascade which shows that *ERT* can reduce the error much faster than other baselines. Note that we have also provided the results of running *EF+CB* multiple times and taking the median of final predictions. The results show that similar error rate to *EF+CB* can be achieved by our method with an order of magnitude less computation.

We have also provided results for the widely used LFPW[1] dataset (Table 2). With our *EF+CB* baseline we could not replicate the numbers reported by [2]. (This could be due to the fact that we could not obtain the whole dataset.) Nevertheless our method surpasses most of the previously reported results on this dataset taking only a fraction of the computational time needed by any other method.

	[1]	[2]	EF	EF+CB	EF+CB (5)	EF+CB (10)	ERT
Error	.040	**.034**	.051	.046	.043	.041	.038

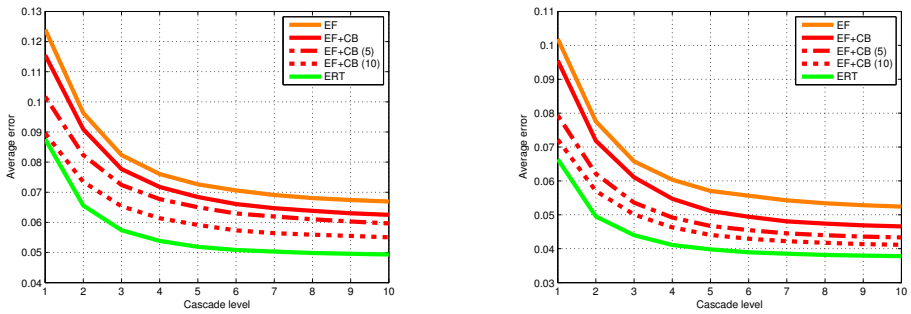
Table 2. A comparison of the different methods when applied to the LFPW dataset. Please see the caption for table 1 for an explanation of the numbers.

Feature Selection: Table 4 shows the effect of using equation (12) as a prior on the distance between pixels used in a split instead of a uniform prior on the final results. The parameter λ determines the effective maximum distance between the two pixels in our features and was set to 0.1 in our experiments. Selecting this parameter by cross validation when learning each strong regressor, r_t, in the cascade could potentially lead to a more significant improvement. Figure 4 is a visualization of the selected pairs of features when the different priors are used.

	Uniform	Exponential
Error	.053	.049

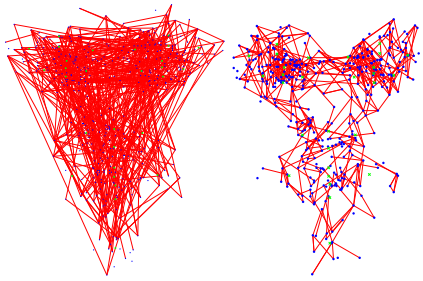
Table 3. The effect of using different priors for feature selection on the final average error. An exponential prior is applied on the Euclidean distance between the two pixels defining a feature, see equation (12).

Regularization: When using the gradient boosting algorithm one needs to be careful to avoid overfitting. To obtain lower test errors it is necessary to perform some form of regularization. The simplest approach is shrinkage. This



(a) HELEN (b) LFPW

Figure 3. A comparison of different methods on HELEN(a) and LFPW(b) dataset. *EF* is the ensemble of randomized ferns and *EF+CB* is the ensemble of ferns with correlation based feature selection initialized with the mean shape. We also provide the results of taking the median of results of various initializations (5 and 10) as suggested by [2]. The results show that the proposed ensemble of regression trees (*ERT*) initialized with only the mean shape consistently outperforms the ensemble of ferns baseline and it can reach the same error rate with much less computation.



(a) Uniform prior (b) Exponential prior

Figure 4. Different features are selected if different priors are used. The exponential prior biases the selection towards pairs of pixels which are closer together.

involves setting the learning rate ν in the gradient boosting algorithm to less than 1 (Here we set $\nu = 0.1$). Regularization can also be achieved by averaging the predictions of multiple regression trees. This way, g_k correspond to a random forest as opposed to one tree and we set $\nu = 1$. Therefore, at each iteration of the gradient boosting algorithm instead of fitting one regression tree to the residuals, we fit multiple trees (10 in our experiments) and average the results. (The total number of trees is fixed in all the cases.)

In terms of the bias and variance trade off, the gradient boosting algorithm always decreases the bias but increases the variance. But regularizing by shrinkage or averaging effectively reduces the variance by learning multiple overlapping models.

	Unregularized	Shrinkage	Averaging
Error	.103	.049	.049

Table 4. A comparison of the results on the HELEN dataset when different forms of regularization are applied. We found similar results when using either shrinkage or averaging given the same total number of trees in the ensemble.

We achieved similar results using the averaging regularization compared to the more standard shrinkage method. However, regularization by averaging has the advantage of being more scalable, as it enables parallelization during training time which is especially important for solving large scale problems.

Cascade: At each level of the cascade the second level regressors can only observe a fixed and sparse subset of the shape indexed features. Indexing the features based on the current estimate is a crude way of warping the image with a small cost. Table 5 shows the final error rate with and without using the cascade. We found significant improvement by using this iterative mechanism which is in line with previously reported results [8, 2] (For a fair comparison here we fixed the total number of observed features to 10×400 points.)

# Trees	1×500	1×5000	10×500
Error	.085	.074	.049

Table 5. The above results show the importance of using a cascade of regressors as opposed to a single level ensemble.

Training Data: To test the performance of our method with respect to the number of training images, we trained different models from differently sized subsets of the training data. Table 6 summarizes the final results and figure 5 is a plot of the error at each level of the cascade. Using many levels of regressors is most useful when we have a large number of training examples.

We repeated the same experiments with the total number of augmented examples fixed but varied the combination of initial shapes used to generate a training example from one labelled face example and the number of annotated images used to learn the cascade (Table 7).

Augmenting the training data using different initial

# Examples	100	200	500	1000	2000
Error	.090	.074	.059	.054	.049

Table 6. Final error rate with respect to the number of training examples. When creating training data for learning the cascade regressors each labelled face image generated 20 training examples by using 20 different labelled faces as the initial guess for the face's shape.

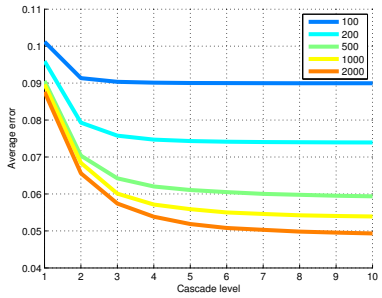


Figure 5. The average error at each level of the cascade is plotted with respect to number of training examples used. Using many levels of regressors is most useful when the number of training examples is large.

# Examples	100	200	500	1000	2000
# Initial Shapes	400	200	80	40	20
Error	.062	.057	.054	.052	.049

Table 7. Here the effective number of training examples is fixed but we use different combinations of the number of training images and the number of initial shapes used for each labelled face image.

shapes expands the dataset in terms of shape. Our results show this type of augmentation does not fully compensate for a lack of annotated training images. Though the rate of improvement gained by increasing the number of training images quickly slows after the first few hundred images.

Partial annotations: Table 8 shows the results of using partially annotated data. 200 training examples are fully annotated and the rest are only partially annotated.

# Examples	200	200+1800(25%)	200+1800(50%)	2000
Error	.074	.067	.061	.049

Table 8. Results of using partially labelled data. 200 examples are always fully annotated. The values inside the parenthesis show the percentage of landmarks observed.

The results show that we can gain substantial improvement by using partially labelled data. Yet the improvement displayed may not be saturated because we know that the underlying dimension of the shape parameters are much lower than the dimension of the landmarks (194×2). There is, therefore, potential for a more significant improvement with partial labels by taking explicit advantage of the correlation between the position of landmarks. Note that the gradient boosting procedure described in this paper does not take advantage of the correlation between landmarks. This issue could be addressed in a future work.

4. Conclusion

We described how an ensemble of regression trees can be used to regress the location of facial landmarks from a sparse subset of intensity values extracted from an input image. The presented framework is faster in reducing the error compared to previous work and can also handle partial or uncertain labels. While major components of our algorithm treat different target dimensions as independent variables, a natural extension of this work would be to take advantage of the correlation of shape parameters for more efficient training and a better use of partial labels.

Acknowledgements: This work has been funded by the Swedish Foundation for Strategic Research within the project VINST.

References

[1] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In *CVPR*, pages 545–552, 2011. 1, 5

[2] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *CVPR*, pages 2887–2894, 2012. 1, 2, 3, 4, 5, 6

[3] T. F. Cootes, M. Ionita, C. Lindner, and P. Sauer. Robust and accurate shape model fitting using random forest regression voting. In *ECCV*, 2012. 1

[4] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995. 1, 2

[5] D. Cristinacce and T. F. Cootes. Boosted regression active shape models. In *BMVC*, pages 79.1–79.10, 2007. 1

[6] M. Dantone, J. Gall, G. Fanelli, and L. V. Gool. Real-time facial feature detection using conditional regression forests. In *CVPR*, 2012. 1

[7] L. Ding and A. M. Martínez. Precise detailed detection of faces and facial features. In *CVPR*, 2008. 1

[8] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In *CVPR*, pages 1078–1085, 2010. 1, 2, 6

[9] G. J. Edwards, T. F. Cootes, and C. J. Taylor. Advances in active appearance models. In *ICCV*, pages 137–142, 1999. 1, 2

[10] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer-Verlag, 2001. 2, 3

[11] V. Kazemi and J. Sullivan. Face alignment with part-based modeling. In *BMVC*, pages 27.1–27.10, 2011. 2

[12] V. Le, J. Brandt, Z. Lin, L. D. Bourdev, and T. S. Huang. Interactive facial feature localization. In *ECCV*, pages 679–692, 2012. 5

[13] L. Liang, R. Xiao, F. Wen, and J. Sun. Face alignment via component-based discriminative search. In *ECCV*, pages 72–85, 2008. 1

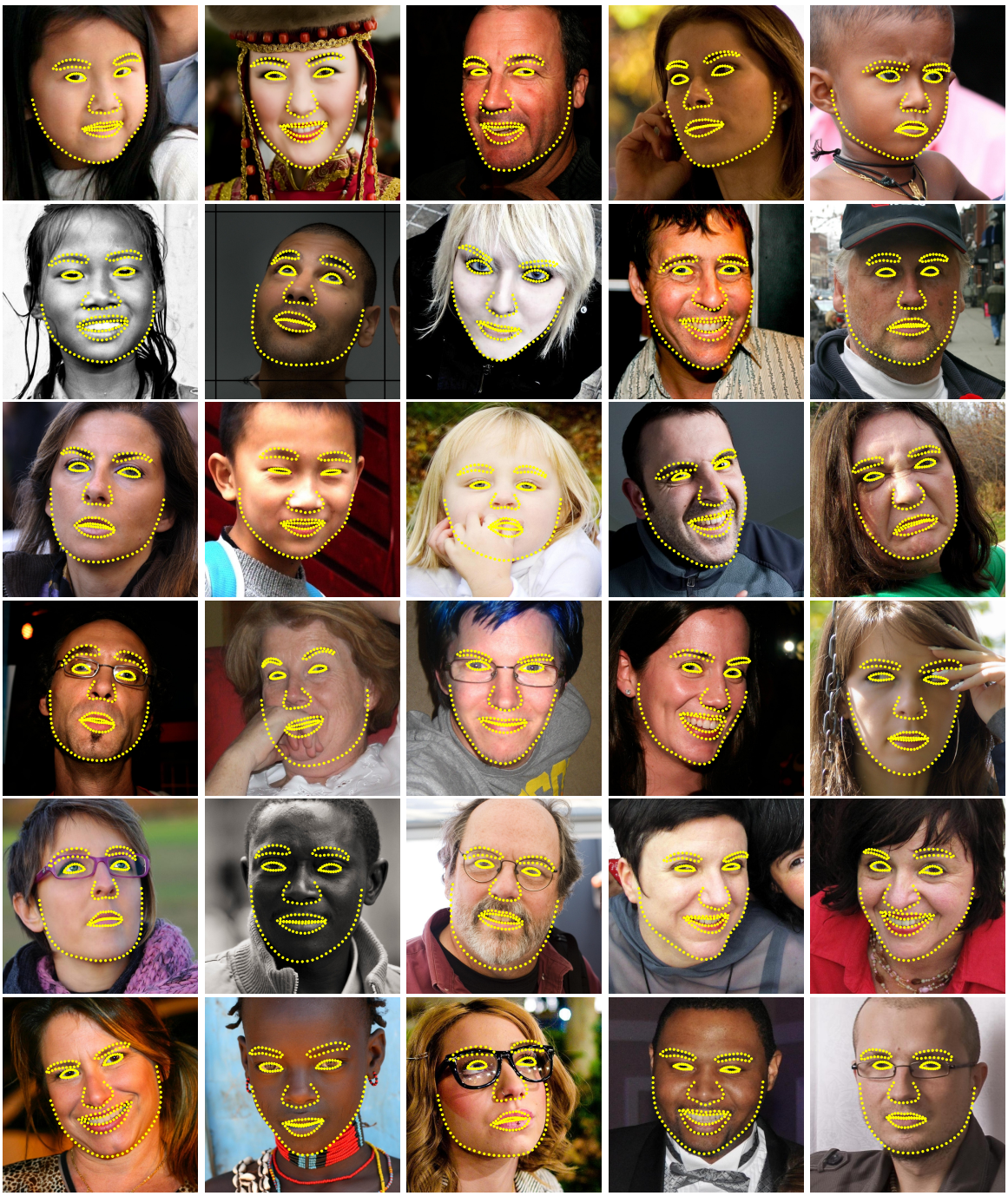


Figure 6. Final results on the HELEN database.

[14] S. Milborrow and F. Nicolls. Locating facial features with an extended active shape model. In *ECCV*, pages 504–513, 2008. 5

[15] J. Saragih, S. Lucey, and J. Cohn. Deformable model fitting by regularized landmark mean-shifts. *Internation Journal of Computer Vision*, 91:200–215, 2010. 1

[16] B. M. Smith and L. Zhang. Joint face alignment with non-parametric shape models. In *ECCV*, pages 43–56, 2012. 1

[17] P. A. Viola and M. J. Jones. Robust real-time face detection. In *ICCV*, page 747, 2001. 5

[18] X. Zhao, X. Chai, and S. Shan. Joint face alignment: Rescue bad alignments with good ones by regularized re-fitting. In *ECCV*, 2012. 1

[19] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, pages 2879–2886, 2012. 1